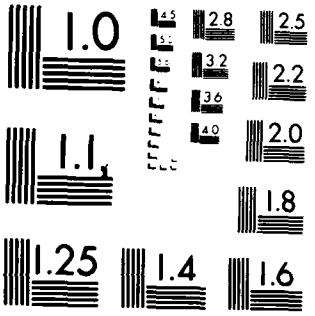
AD-A131 085 DACS (DATA AND ANALYSIS CENTER FOR SOFTWARE) CONVERSION 1/1
DATA COLLECTION FORMS(U) DATA AND ANALYSIS CENTER FOR
SOFTWARE GRIFFISS AFB NY JUN 81 F30602-78-C-0255

UNCLASSIFIED F/G 14/2 NL

END
DATE FILMED
8 83
DTIC




MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: /
2. GOVT ACCESSION NO.: AD-A131 085
3. RECIPIENT'S CATALOG NUMBER:

4. TITLE (and Subtitle): DACS CONVERSION DATA COLLECTION FORMS

5. TYPE OF REPORT & PERIOD COVERED: Interim Report Jan. 1981 - June 1981

6. PERFORMING ORG. REPORT NUMBER: N/A

7. AUTHOR(s): N/A

8. CONTRACT OR GRANT NUMBER(s): F30602-78-C-0255

9. PERFORMING ORGANIZATION NAME AND ADDRESS: Data & Analysis Center for Software RADC/ISISI Griffiss AFB, NY 13441

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:

11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (COEE) Griffiss AFB, NY 13441

12. REPORT DATE: June 1981

13. NUMBER OF PAGES: 17

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same

15. SECURITY CLASS. (of this report): UNCLASSIFIED

15a. DECLASSIFICATION DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: John Palaimo (COEE)
DACS Source Code No. 413570
Cost: [redacted]

Available from:
Data & Analysis Center for Software
RADC/ISISI
Griffiss AFB, NY 13441

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Software Conversion
Conversion Costs
Data Collection
Software Experience Data
Data Parameters
Data Repository

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This product consists of three data collection forms to be used to describe conversion projects. The first form, the Software System Overview Form is designed to capture data which describes the general nature of the conversion effort. The second form, the Detailed Resource Expenditure Form, is used to capture data on the personnel effort and machine usage required to perform the conversion. The Conversion Problem Report Form is used to identify problems encountered, as well as the method of detecting and correcting errors. The set contains 3 forms and 13 pages of definitions and guidelines for using the forms.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

# DACS CONVERSION DATA COLLECTION FORMS

## Purpose

The economics and logistics of software inventory conversion are major procurement issues. In many cases the cost of the software conversion equals or exceeds the cost of the hardware procured. When a large multi-site hardware upgrade is involved, conversion may take years and require a sizeable dedicated programming staff to complete. Conversion is a relatively new area of interest to software engineers but one that promises to present a whole new range of problems. At this time, little is known about the types of problems likely to occur during a conversion, the frequency with which they occur across projects, their severity, and how they may be corrected or prevented. By collecting data on past and present conversion efforts, information can be compiled which can be used for identifying problems and their demographics. A database of conversion data will help to conduct feasibility studies, estimate costs for performing conversions, identify conversion cost drivers, and help to establish cost-benefit relationships for conversion aids and tools. The DACS is seeking to establish such a database which will be made available to the software engineering community.

## The Conversion Data Collection Forms

This document describes

DACS ~~has designed~~ designed three data collection forms to be used during software conversion. The first form, the Software System Overview (SSO) seeks to capture those types of data which describe the general nature of the conversion effort. The second form, the Detailed Resource Expenditure (DRE) form, seeks to capture data which could be used to identify the personnel effort and machine usage required to perform the conversion. (Figure 1 graphically depicts a sequence of activities which occur during a conversion.) The third form, the Conversion Problem Report (CPR) is used to identify problems encountered, as well as the method of detecting and correcting errors. By filling out a CPR as soon as a problem surfaces during a conversion effort, the chances of any modification being overlooked can be minimized.

---

Three reports were used extensively in preparing these forms. They are:

(1) DPSC Experience Using the NAVDAC Conversion Management System prepared by GSG, Inc., 51 Main St., NH, March 10, 1980.

(2) Handbook for Estimating Conversions Costs of Large Business Programs by Paul Oliver, director of the Federal Compiler Testing Service, Washington, DC 20376, Feb. 14, 1979.

(3) The Software Engineering Laboratory by Victor R. Basili, et al., Technical Report TR-535, SEL 1, May 1977. 104 p. Available from: V. R. Basili, University of Maryland, College Park, MD.

Software System Overview (SSO)

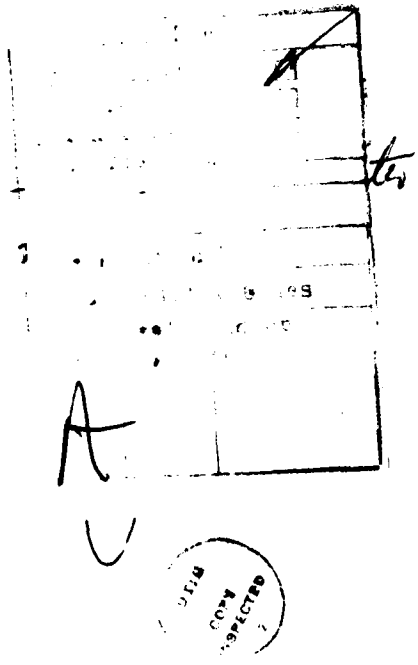
This form is used to collect data concerning (1) the general nature of the conversion effort, (2) the size and characteristics of the software system before the conversion and after, (3) the difficulty of the conversion and (4) the technology utilized.

Detailed Resource Expenditure (DRE)

This form is used to collect data concerning the effort required to perform the subtasks required during a conversion with respect to personnel and machine resources. Much of the data on this form will be a summary of the Conversion Problem Reports involved in the conversion effort.

Conversion Problem Report (CPR)

This form can be used as a quality assurance tool and tracking mechanism for the conversion effort. The completion of a CPR each time a new problem surfaces creates a formal recording and tracking machanism. It is recomended that a lower level or installation-oriented worksheet be employed to keep a record of each detail in the conversion effort related to this CPR. At or near termination of the change/correction effort, the detailed data can be transcribed to the CPR form.



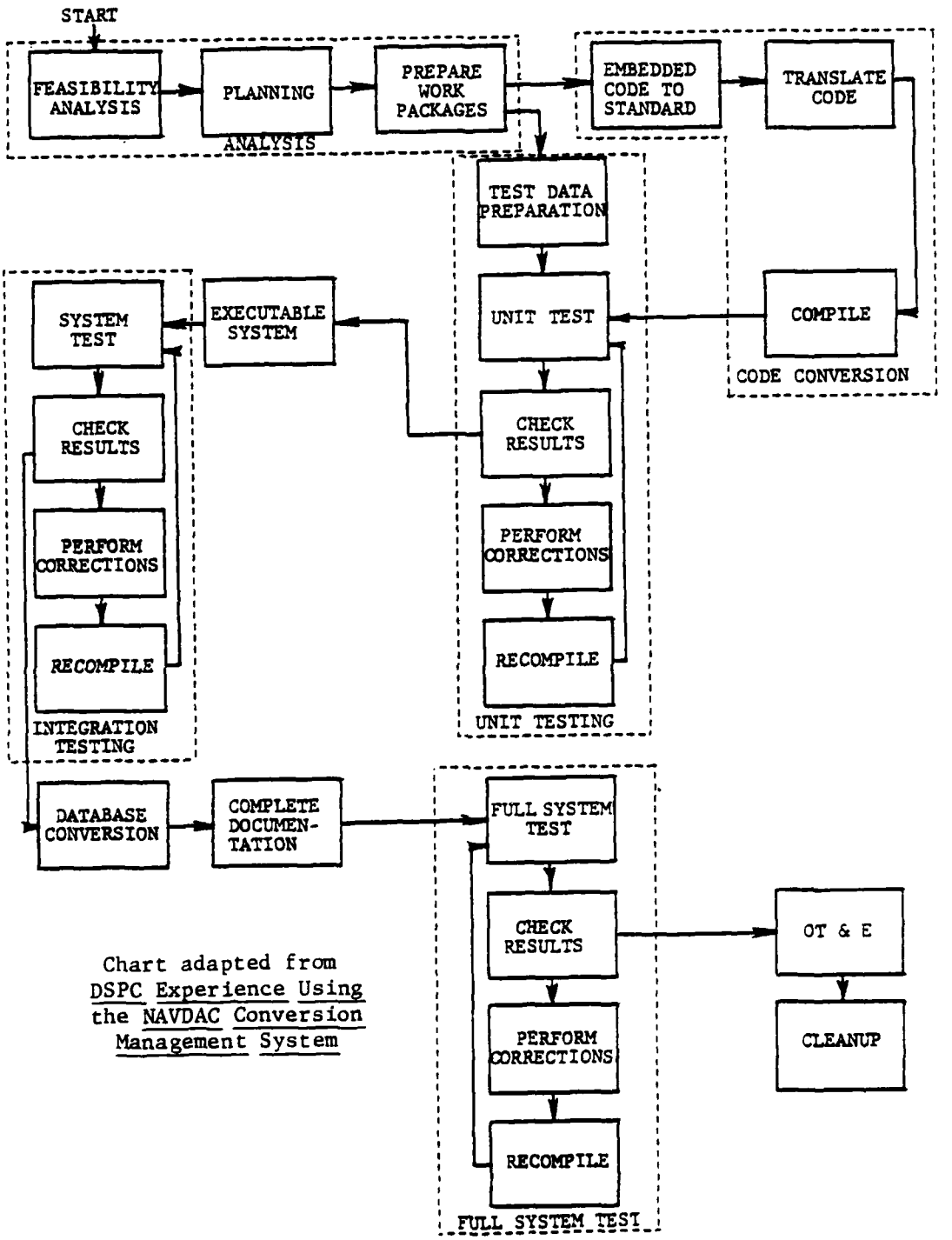



Chart adapted from DSPC Experience Using the NAVDAC Conversion Management System

FIGURE 1 CONVERSION TASKS

DACS CONVERSION DATA COLLECTION FORMS

SYSTEM OVERVIEW

INSTRUCTIONS

The purpose of the system overview is to describe the conversion environment. This form should be completed at both the beginning and at the end of the conversion process. The goal is to have a description of the software system characteristics before and after conversion.

Explanations and definitions of the form's data elements are presented below.

1. Project Name ID - Give the name of the project or a special ID assigned to the project.

2. System Description - Give a brief summary of the application of the system software being converted.

3. System/CPCI's Name - The name(s) of the highest level system components being converted.

NUMBER OF SYSTEM COMPONENTS

Give the number of components as they apply to the software system being converted.

4. CPCI - Computer Program Configuration Item - This is a software system which is executed separately. It is the highest level of executable software which does not cross computer boundries. For example, an operating system and an application program are separate CPCI's.

5. CPCG - Computer Program Configuration Group - This is one level lower than a CPCI. It is generally structured along functional lines. For example, separate links of a multiple overlay structure may be separate CPCGs. Different CPCGs within a CPCI are callable programs which perform major, non-overlapping functions.

6. CPC - Computer Program Component - This is one level lower than a CPCG. It does not constitute an executable package. Software at this level is an individual subprogram which calls lower level subprograms. These may contain, INCLUDE or SELECT statements to incorporate separately defined segments (blocks) of source code into the CPC.

7. Segment - The lowest level of source code in a program or CPCI. It is a block of code which evaluates a particular algorithm or peforms a very specific interface function such as COMPOL or LABELED (named) COMMON block(s). It is not a callable subroutine.

8. Conversion Duration - Indicate the total duration of the conversion project in months. Use the letter "E" to indicate if this parameter is estimated.

9. Conversion Type - Note the type of conversion and the specific requirement, (e.g. language, from COBOL 68 to COBOL 74).

SOFTWARE SYSTEM CHARACTERISTICS

The following items describe the software system at the beginning and at the end of the conversion.

10. Language(s) - The language or languages in which the program(s) comprising this software system has(have) been written. This will be the host language at the beginning and the target language at the end of the conversion.

11. Machine - The computer on which the system was developed for or converted to run on. It will be the host machine at the beginning and the target machine at the end of the conversion project.

12. Special Hardware Requirements - Note only those hardware requirements which are going to impact the conversion effort.

13. Number of lines of Code - Indicate how many lines of source code make up the system being converted exclusive of generalized sorts and other system utilities. Indicate whether the number recorded referes to E - Executable Code or A - Actual Lines including comments and other non-executable lines. Do this for both (HOL) High Order Language on (ALC) Assembler Language Code.

14. Number of Job Streams - Job Streams are the number of job/run streams that are used to process a sequence of application programs. Job/run streams typically involve execution of two or more programs, with or without sorts and utilities, executed in sequence without interruption and with data passing from one program to another.

15. Number of Independent Runs - Record the number of independent runs. Independent runs are the number of programs that run independently without updating or maintaining files (e.g. report writer extract from the master file).

16. Number of files - Indicate how many data files and master files are accessed by all of the system component(s). Temporary work-in-process files which are not normally saved should be included if it's anticipated that they may be affected by the conversion.

17. Number of Databases - Indicate the number of databases required by the software system.

18. Support Software Developed/Utilized for This Conversion - List the executable software that was developed and/or used by the project to specifically support the conversion effort. Enter detailed information only if immediately available or if the utilized support software is not a well known product. List such information as source language,

number of lines of code (indicate whether the reference is to executable code). Typical examples are translators, meta-compilers, test drivers and graphics interface software.

19. Resource Expenditure Forms - List all the Detailed Resource Expenditure form numbers which contain the effort expenditures required to perform this conversion.

20. This Form Prepared by - Give the name of the person completing the form.

21. Project Contact - Give the name of the person who should be contacted with respect to acquiring additional data or for clarification of data on this form.

22. Staff Size - Indicate how many people are assigned to the project. Include administrative, technical and clerical personnel.

23. Conversion Performed - Please check those descriptors which apply to who performed the conversion and where it was performed.

24. Difficulty of Conversion - Give the number of system components or modules belonging to the system which are of each level of difficulty.
    Translation - Essentially automated (at least 90% of source code) conversion with little personnel intervention during the translation process.
    Complex Translation - Conversion is largely automated (75 - 90% of source code) but logic changes are required and must be performed .
    Redesign or reprogramming - Automated translation is applicable to no more than 75% of the source code (but at least 50% can be translated); significant logic and database changes are necessary.

25. Special Considerations/Constraints - Note any environmental or other factors which impact the conversion effort (e.g. limited availability of machine resources, such as, memory/CPU usage, peripheral device(s) or other hardware differences or requirements changes during conversion).

26. Technology - Note the software engineering technologies and techniques applicable to the project. Technologies not listed may be added (Appendix 1 defines the listed technologies). In the space preceding the Technology, check under 'D' if the technology was utilized during original development (if that information is available). Check under 'C' if the technology was utilized during conversion. Check under both D and C if the technology was utilized during both development and conversion. Check under 'T' is the technology was utilized only during testing.

DACS
RADC/ISISI
Griffiss AFB, NY 13441
315/336-0937
Data & Analysis Center for Software

# CONVERSION DATA COLLECTION FORMS

## SYSTEM OVERVIEW

PROJECT NAME ID: _____

SYSTEM DESCRIPTION: _____

SYSTEM/CPCI NAME(S): _____

NUMBER OF SYSTEM COMPONENTS (to date):

CPCIs _____ CPCGs _____

CPCs _____ Segments _____

CONVERSION DURATION:
(E-Estimated; A-Actual)

_____

CONVERSION TYPE:

Language ___ Machine ___

SOFTWARE SYSTEM CHARACTERISTICS
(Indicate D-Development; C-Conversion if known and or where it may be appropriate)

Language(s): _____

Machine (make/model): _____

Special Hardware Requirements (such as):

Main Frame Memory: _____

Card/Paper Tape Readers/Punch: _____

Tape Drives: _____ Plotters: _____

On-Line Graphics: _____

Other: _____

Number of Lines of Code: (HOL) _____ (ALC) _____

Number of Job Streams: _____

Number of Independent Runs: _____

Number of Files: Temporary Permanent

Before Conversion: _____ _____

After Conversion: _____ _____

Number of Databases: _____

SUPPORT SOFTWARE DEVELOPED/UTILIZED FOR THIS CONVERSION:

(Indicate whether existing software was used-E or whether special software was developed-D)

| Name | Language | Lines | Function | E/D |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |

RESOURCE EXPENDITURE FORMS:
(Utilized on this project to date)

_____ _____ _____

_____ _____ _____

_____ _____ _____

THIS FORM PREPARED BY: _____ DATE: _____

PROJECT CONTACT:

Name/Title: _____

Address: _____

_____ Telephone: _____

STAFF SIZE: _____

CONVERSION PERFORMED: (Check all that apply)

___ At Customer Site
___ In-House
___ At Contractor Site
___ By Customer
___ By In-House Personnel
___ By Contractor
___ For One-Site System
___ For Multi-Site System
___ Other _____

DIFFICULTY OF CONVERSION:

| TYPE | NUMBER OF PROGRAM COMPONENTS AFFECTED | | | | | |
|---|---|---|---|---|---|---|
| | CPCIs | CPCGs | CPCs | Segments | LINES HOL | LINES ALC |
| Translation | | | | | | |
| Complex Translation | | | | | | |
| Redesign or Reprogramming | | | | | | |

SPECIAL CONSIDERATIONS/CONSTRAINTS: _____

_____

_____

TECHNOLOGY/TOOLS UTILIZED (to date):
(D-Original Development, C-Conversion Effort, T-Testing Effort)

| D C T | | D C T | |
|---|---|---|---|
| ( )( )( ) | Reusable Code | ( )( )( ) | Simulation |
| ( )( )( ) | Correctness Proofs | ( )( )( ) | Structured Programming |
| ( )( )( ) | Code Standards Auditor | ( )( )( ) | Walk-Through |
| ( )( )( ) | Consistency Checker | ( )( )( ) | Critical Piece First |
| ( )( )( ) | Independent Test Team | ( )( )( ) | Database Analyzer |
| ( )( )( ) | Chief Programmer Team | ( )( )( ) | Data Dictionary |
| ( )( )( ) | Automated Design Tools | ( )( )( ) | Documentation Generator |
| ( )( )( ) | Automated Requirements Tools | ( )( )( ) | High Order Language |
| ( )( )( ) | HIPO Design Aides | ( )( )( ) | Pre-Compiler |
| ( )( )( ) | Process Design Language | ( )( )( ) | Program Flow Analyzer |
| ( )( )( ) | Structure Charts | ( )( )( ) | Meta-Compiler |
| ( )( )( ) | Top-Down Development | ( )( )( ) | Meta-Language |
| ( )( )( ) | Modular Decomposition | ( )( )( ) | Translator _____ |
| ( )( )( ) | Program Support Library | ( )( )( ) | Emulator _____ |
| ( )( )( ) | Design Review | ( )( )( ) | Auditor _____ |
| ( )( )( ) | _____ | ( )( )( ) | _____ |

DACS CONVERSION DATA COLLECTION FORMS

DETAILED RESOURCE EXPENDITURE

INSTRUCTIONS

The purpose of this form is to collect resource expenditure data at the subtask level. Much of the data for this form is accumulated from the Conversion Problem Report(s) (CPRs) involved in the conversion effort. Several of these forms may be collected on a large project (where different "teams" might be assigned to different groups of Conversion Problem Reports). Make out at least one of these forms for each CPCI comprising the software system being converted. In cases where data is unavailable or not appropriate to the project, indicate with UN = Unavailable or NA = Not appropriate.

1. Project Name - Give the name of the project.

2. CPCI Name - Give the name of the CPCI (program).

3. CPCI Description - Give a brief summary of the function or type of application of this CPCI.

4. Date Start/Finish - Indicate the date of the beginning and the end of the conversion effort for this CPCI.

5. Form Prepared By - The name of the person initiating the form.

6. Contact/Team Leader - Give the name of the person to be contacted for further information about the resources recorded on this form. This contact person may not necessarily be the same as the one named on the System Overview Form.

7. CPRs - List the Conversion Problem Report form numbers which are summarized on this form.

8. Structures Impacted - Enter the number of relevent components (CPCGs, CPCs, segments, lines, files, databases, etc.) of the CPCI (program) which have been modified, deleted and added. Each of these components have been defined in the System Overview instructions.

9. Feasibility Analysis - Give the personnel and computer usage effort data to perform the feasibility analysis. This is the analysis required in order to determine the best way of replacing an existing data processing system with a new one. Activities include defining the inadequacies of the current system and collecting an inventory of the current software products, defining how the new system should differ from the old one and providing recommendations for converting to the new system.

10. Prepare Conversion Specifications - Give the personnel and computer usage effort data to prepare the conversion specifications and to develop a detailed description of work to be done. The "specifications"

document the results of the feasibility analysis and are used as a master plan for implementing the conversion.

11. Generate and Validate Test Data - Give the personnel and computer usage effort data required to generate and validate the test data. This includes selecting and generating sample test data and determing the effectiveness with which the test data is able to exercise programs.

12. Recode Embedded Code - Give the personnel and computer usage effort data for recoding embedded code. This involves recoding to standard Higher Order Language or to the Assembler Level Code of the target machine.

13. Translate to Target Language/Machine - Give the personnel and computer usage effort data for editing and transforming higher order language programs to obtain compatibility with the target systems compiler.

14. Perform Unit Testing - Give the personnel and computer usage effort data for individually testing each converted subroutine.

15. Perform Integration Testing - Give the personnel and computer usage effort data for constructing an executable version of the subsystem and completing integration testing. This includes replacing the modules which were modified with tested modules.

16. Convert Database - Give the personnel and computer usage effort data for converting production file media and formating to the target language/machine.

17. Complete Documentation - Give the personnel and computer usage effort data for completing the final documentation to identify changes, developing a system test plan and developing an updated user's guide.

18. Perform Full System Test - Give the personnel and computer usage effort data to perform full system testing. This includes replacing all subsystems which were modified and tested and achieving successful production runs using actual or simulated test data.

19. Perform Operational Test and Evaluation - Give the personnel and computer usage effort data for operational testing and evaluation. This involves running a redundant production operation using the new configuration with the goal of final user acceptance. Actual data from the program's operational environment is usually utilized at this stage.

20. Clean-up - Give the personnel and computer usage effort data for clean-up (e.g. construct library tapes, archive old programs, etc.).

(Figure 1 shows the sequential relationship of the individual tasks involved in an overall conversion of a system.)

PERSONNEL (MAN-HOURS)

21. Manager - Note the number of hours of management-level activity charged to each task.

22. Conversion Analyst - Note the number of hours charged to each task. These tasks include feasibility analysis and requirements analysis.

23. Technical - Note the number of hours charged to each task. These tasks include design, programming and testing.

24. Clerical - Note the number of clerical hours charged to each task. Clerical activities include keypunching, data entry, typing documentation, submitting runs and maintaining the program library.

COMPUTER USAGE

25. Machine - Note the name and model of the machine (e.g. IBM 370, Honeywell 6180, etc.)

26. No. Runs - Note the total number of computer runs to complete each task.

27. Computer Time - Note the total number of CPU and I/O hours to complete each task.

28. Completed By - Note the name of the individual completing the entries on this line.

29. Support Software Developed/Utilized for This Effort - List the executable software that was developed and/or used by the project to specifically support this conversion effort. Enter detailed information only if immediately available or if the utilized support software is not a well known product. List such information as source language, number of lines of code (indicate whether the reference is to executable code). Typical examples are translators, meta-compilers, test drivers and graphics interface software.

DACS
RADC/ISISI
Griffiss AFB, NY 13441
315/336-0937
Data & Analysis Center for Software

CONVERSION DATA COLLECTION FORMS

DETAILED RESOURCE EXPENDITURE

RESOURCE EXPENDITURE FORM NUMBER: _____

PROJECT NAME: _____

CPCI NAME: _____

CPCI DESCRIPTION: _____

DATE START: _____ DATE FINISH: _____

CONTACT/TEAM LEADER: _____

FORM PREPARED BY: _____

CPRs: _____ _____ _____

_____ _____ _____

_____ _____ _____

| STRUCTURES IMPACTED | MODIFIED | DELETED | ADDED |
|---|---|---|---|
| Number of CPCGs | | | |
| Number of CPCs | | | |
| Number Segments | | | |
| Number Lines | | | |
| Number Temporary files | | | |
| Number Permanent files | | | |
| Number *Databases* | | | |
| Number Subsystems | | | |
| Other | | | |

| CONVERSION TASK DESCRIPTION | PERSONNEL (MANHOURS) | | | | COMPUTER USAGE | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Manager | Conversion Analyst | Technical | Clerical | Machine Name | No. Runs | Computer Time (hours) CPU | Computer Time (hours) I/O | Completed by |
| Feasibility Analysis | | | | | | | | | |
| Prepare Conversion Specifications | | | | | | | | | |
| Generate and Validate Test Data | | | | | | | | | |
| Recode Embedded Code (if any) | | | | | | | | | |
| Translate to Language/Machine | | | | | | | | | |
| Perform Unit Testing | | | | | | | | | |
| Perform Integration Testing | | | | | | | | | |
| Convert *Database(s)* | | | | | | | | | |
| Complete Documentation | | | | | | | | | |
| Perform Full System Test | | | | | | | | | |
| Perform Operational Test and Evaluation | | | | | | | | | |
| Cleanup | | | | | | | | | |

SUPPORT SOFTWARE DEVELOPED/UTILIZED FOR THIS EFFORT: (Indicate special purpose/general purpose - S/G)

| Support Software Developed for This Effort | | | | Existing Support Software Used for This Effort | | | |
|---|---|---|---|---|---|---|---|
| Lines of Code | Languages(s) | Function | S/G | Lines of Code | Languages(s) | Function | S/G |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

DACS CONVERSION DATA COLLECTION FORMS

CONVERSION PROBLEM REPORT

INSTRUCTIONS

The purpose of this form is to capture data which can be used for analysis of the types of problems which occur during a conversion as well as to identify the most beneficial methods of detection and correction of these problems/faults. Complete a separate form for each problem encountered during the conversion.

1. Project - Give the name of the project.

2. Number - This number is to be assigned by the reviewer in charge of tracking the conversion effort. The number should be a unique number or code so as to uniquely identify this conversion problem effort. It is suggested that the numbers be issued in sequence and that the sequence be preceded by two digits representing the year the problem surfaced.

3. Current date - Indicate the date on which an entry is first made on the form, even if the form is not completed at that time.

4. Date Start - Indicate the date when it was first realized a problem existed.

5. Date Finished - Indicate the date when the problem was corrected, worked around or otherwise closed.

6. CPCI/CPCG/CPC/Segment - Circle the level of the highest system component being impacted by this problem and specify the name of that component.

7. Contact/Team Leader - Give the name, address and telephone number(s) of the person who should be contacted with respect to acquiring additional data or for classification of data on this form.

8. How was the Problem Detected - Indicate what event triggered an awareness of the problem.

9. Nature of Problem - Give a brief description of the problem in terms of the functional application.

10. Category of Problem/Fault - Check the category which applies. If the category is not listed, supply a category in the space provided which corresponds to the description in item 9. More than one item may be checked.

11. Change Needed to Correct Problem - Give a brief description of the change needed to correct the problem.

TIME TO:

12. Isolate Error - Indicate the number of hours required to determine the source and scope of the error.

13. Design Change - Indicate the number of hours required to design, not implement the change.

14. Implement Change - Indicate the number of hours required to implement the change, including recoding, integration, testing and operational implementation.

15. Total Number of CPCGs, CPCs, Segments, Lines, Files, Databases, etc. - Enter the number of the software system components at each level of detail which have been modified, deleted and/or added to resolve the problem/fault being addressed by this CPR.

16. Techniques/Tools used in Detecting the Fault, Isolating the Cause and Designing and/or Implementing Change - Mark an "S" or a "T" in all cells which apply to this problem/fault. Use an "S" to indicate that this tool or technique was Successful or Marginally Succesful in achieving the goal; use a "T" to indicate that the tool or technique was Tried. Enter the total number of manhours and computer hours used during each listed activity.

17. Additional Information - This section is intended to permit explanation of any items you feel may be significant in categorizing the problem or fault, understanding its cause, determining how it was found and explaining any effects it may have that are not fully covered in previous sections.

18. Relation to Correction Activities for a Previous Fault/Problem
Yes - If you can determine that the problem was the result of some previous change to the software during this or a previous conversion effort, whether or not in the same subroutine, module or data file. Check this space and reference the problem report form completed for the previous change. Specify if previous change was for enhancement, correction or conversion. Specify if error was detected during analysis, code translation, recoding, unit test, integration test or system test.
No - If you can determine that this problem is unrelated to previous changes made during this conversion, check this space.
Can't tell - If you suspect that this error is related to some previous change during this conversion, but can't be sure, check this space and explain why you are not sure on the line below.

19. Person Filling out this Form - Give the name(s) and telephone numbers of the person(s) filling out this form.

DACS
RADC/ISISI
Griffiss AFB, NY 13441
315/336-0937
Data & Analysis Center for Software

CONVERSION DATA COLLECTION FORMS

CONVERSION PROBLEM REPORT

PROJECT NAME: _____ CONVERSION TYPE: _____ CONVERSION PROBLEM REPORT NUMBER: _____

CPCI/CPCG/CPC/Segment: _____ DATE START: _____ DATE FINISH: _____ CURRENT DATE: _____

CONTACT/TEAM LEADER: _____

HOW WAS THE PROBLEM DETECTED? _____

_____

_____

NATURE OF THE PROBLEM (Description): _____

_____

_____

CATEGORY OF PROBLEM/FAULT:

__ Packing of Data
__ Execution Time Overlays
__ Incompatible Word Length
__ Inconsistent Use of Variable
__ Data Alignment Error
__ Loss of Local Variable Contents
__ Use of Nonstandard Programming Techniques/Practices
__ Nonstandard Language Extensions Used
__ Nonstandard Argument Convention
__ Storage Incompatibilities
__ Misallocation of Local Storage
__ Array Indexing Out of Range
__ Use of Internal/Intrinsic Function(s) Having Nonstandard Implementation: ___
_____
__ Original Design Specification Requirements Not Met: _____
_____
__ Programming Error(s) Not Detected Until Conversion Effort: _____
_____
__ Software Interface: _____
__ Other: _____
_____

CHANGE NEEDED TO CORRECT PROBLEM: _____

_____

_____

TIME TO (HOURS):
ISOLATE ERROR/IDENTIFY CHANGE: _____ DESIGN CHANGE: _____ IMPLEMENT CHANGE: _____

| | MODIFIED | DELETED | ADDED | | MODIFIED | DELETED | ADDED |
|---|---|---|---|---|---|---|---|
| Number of CPCGs | | | | Number Temporary Files | | | |
| Number of CPCs | | | | Number Permanent Files | | | |
| Number Segments | | | | Number Databases | | | |
| Number Lines | | | | Other | | | |

TECHNIQUES/TOOLS USED IN DETECTING THE FAULT OR ISOLATING THE CAUSE AND DESIGNING AND/OR IMPLEMENTING CHANGES:
(enter T for Tried and/or S for Successful where appropriate)

| Resources/Activities | Detection | Isolating Cause | Implementing Change | Testing Change | Man Hours | Machine Hours |
|---|---|---|---|---|---|---|
| Design Review | | | | | | |
| Test Runs | | | | | | |
| Code Reading By: | | | | | | |
| Documentation Reading: (Specify which level of documentation) | | | | | | |
| Proof Technique: | | | | | | |
| Trace: | | | | | | |
| Dump | | | | | | |
| Cross-Reference | | | | | | |
| Attribute List | | | | | | |
| Special Debug Techniques | | | | | | |
| Error Messages | | | | | | |
| Inspection of Output | | | | | | |
| Translator: | | | | | | |
| Auditor: | | | | | | |
| Emulator: | | | | | | |
| Converter: | | | | | | |
| Simulator: | | | | | | |
| Automatic Verifier/Program Path Analyzer: | | | | | | |
| Developed Support Software: | | | | | | |
| Other: | | | | | | |
| Other: | | | | | | |

ADDITIONAL INFORMATION: _____

WAS THIS PROBLEM RELATED TO A PREVIOUS MODIFICATION EFFORT? YES _____ NO _____ CAN'T TELL _____

IF YES CIRCLE PREVIOUS MODIFICATION EFFORT CATEGORY: CONVERSION ENHANCEMENT CORRECTION OTHER _____

IF THE PROBLEM WAS A RESULT OF A MODIFICATION DURING THIS CONVERSION EFFORT, GIVE THE NUMBER(S) OF THE CONVERSION PROBLEM REPORT WHICH REQUESTED THE CHANGE: _____

PERSON FILLING OUT THIS FORM: _____ TELEPHONE: _____ / _____
(Area Code)

## APPENDIX 1

## TECHNOLOGY DEFINITIONS

1. Chief Programmer Team - A chief programmer team is a structured team of specialists for software development headed by a chief programmer. The team has at its core three members: the chief programmer, the back-up programmer and the secretary/librarian. The rest of the team consists of programmers as required. The team is normally limited to less than ten members.

2. Automated Design Tools - Computer programs used to provide an understandable representation of the software design as it evolves.

3. Automated Requirements Tools - Computer programs which are used to provide a succinct and unambiguous specification of the system based on computer requirements.

4. HIPO Design Aides - A graphical technique that defines each system component by the transformation of its input datasets to its output datasets and also defines the hierarchical relationships between components.

5. Process Design Language - A formal algorithmic specification of a software component.

6. Structure Charts - A graphical technique which illustrates the relationships between the components of a software system.

7. Top-Down Development - A software development approach that identifies major functions to be accomplished, then proceeds from there to an identification of the lesser functions that derive from the major ones.

8. Modular Decomposition - The process of breaking a large program into small modules that perform complete functions.

9. Program Support Library - A software system which provides tools to organize, implement and control software development.

10. Simulation - A computer program that provides the target system with inputs or responses that resemble those that have been provided by the process for the device being simulated.

11. Structured Programming - The activity of programming with a limited set of program constructs.

12. Walk-Throughs - A formal meeting by various numbers of a project for the review of source code and design for technical adequacy and error detection.

13. Critical Piece First - The implementation of the most critical aspects of the system first.

14. Database Analyzer - A computer program that reports information on every usage of data, identifies each program using any data elements and indicates whether the program inputs, uses, modifies or outputs the data element.

TECHNIQUES/TOOLS USED IN DETECTING THE FAULT OR ISOLATING THE CAUSE AND DESIGNING AND/OR IMPLEMENTING CHANGES:
(enter T for Tried and/or S for Successful where appropriate)

| Resources/Activities | Detection | Isolating Cause | Implementing Change | Testing Change | Man Hours | Machine Hours |
|---|---|---|---|---|---|---|
| Design Review | | | | | | |
| Test Runs | | | | | | |
| Code Reading By: | | | | | | |
| Documentation Reading: (Specify which level of documentation) | | | | | | |
| Proof Technique: | | | | | | |
| Trace: | | | | | | |
| Dump | | | | | | |
| Cross-Reference | | | | | | |
| Attribute List | | | | | | |
| Special Debug Techniques | | | | | | |
| Error Messages | | | | | | |
| Inspection of Output | | | | | | |
| Translator: | | | | | | |
| Auditor: | | | | | | |
| Emulator: | | | | | | |
| Converter: | | | | | | |
| Simulator: | | | | | | |
| Automatic Verifier/Program Path Analyzer: | | | | | | |
| Developed Support Software: | | | | | | |
| Other: | | | | | | |
| Other: | | | | | | |

ADDITIONAL INFORMATION: _____

_____

WAS THIS PROBLEM RELATED TO A PREVIOUS MODIFICATION EFFORT? YES _____ NO _____ CAN'T TELL _____

IF YES CIRCLE PREVIOUS MODIFICATION EFFORT CATEGORY: CONVERSION ENHANCEMENT CORRECTION OTHER _____

IF THE PROBLEM WAS A RESULT OF A MODIFICATION DURING THIS CONVERSION EFFORT, GIVE THE NUMBER(S) OF THE CONVERSION PROBLEM REPORT WHICH REQUESTED THE CHANGE: _____

PERSON FILLING OUT THIS FORM: _____ TELEPHONE: _____/_____
(Area Code)


DATE
ILME